IN THE SPECIFICATION

Page 14, after line 11, please substitute the text previously inserted as follows:

MARKUP

BRIEF DESCRIPTION OF THE ~~DRAWING~~ DRAWINGS

Figure 1 shows a system architecture according to the present invention.

Figure 2 shows a schematic diagram of a system architecture according to the present invention.

CLEAN COPY

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 shows a system architecture according to the present invention.

Figure 2 shows a schematic diagram of a system architecture according to the present invention.

Please replace the paragraph commencing on page 13, line 19, and continuing onto page 20, as follows:

MARKUP COPY

The following U.S. Patents, expressly incorporated herein by reference, define aspects of micropayment, digital certificate, and on-line payment systems: 5,930,777 (Barber); 5,857,023 (Demers et al.); 5,815,657 (Williams); 5,793,868 (Micali); 5,717,757 (Micali); 5,666,416 (Micali); 5,677,955 (Doggett et al.); 5,839,119 (Krsul; et al.); 5,915,093 (Berlin et al.); 5,937,394 (Wong, et al.); 5,933,498 (Schneck et al.); 5,903,880 (Biffar); 5,903,651 (Kocher); 5,884,277 (Khosla); 5,960,083 (Micali); 5,963,924 (Williams et al.); 5,996,076 (Rowney et al.); 6,016,484 (Williams et al.); 6,018,724 (Arent); 6,021,202 (Anderson et al.); 6,035,402 (Vaeth et al.); 6,049,786 (Smorodinsky); 6,049,787 (Takahashi, et al.); 6,058,381 (Nelson); 6,061,448 (Smith, et al.); 5,987,132 (Rowney); and 6,061,665 (Bahreman). See also, Rivest and Shamir, "PayWord and MicroMint: Two Simple Micropayment Schemes" (May 7, 1996), expressly incorporated herein by reference; Micro PAYMENT transfer Protocol (MPTP) Version 0.1 (22-Nov-95) et seq, ~~http://www.w3.org/pub/WWW/TR/WD-mptp~~; Common Markup for web Micropayment Systems, ~~http://www.w3.org/TR/WD-Micropayment-Markup~~ (09-Jun-99); "Distributing Intellectual Property: a Model of Microtransaction Based Upon Metadata and Digital Signatures", Olivia, Maurizio~~, http://olivia.modlang.denison.edu/~olivia/RFC/09/~~.

CLEAN COPY

Paragraph commencing on page 13, line 19:

The following U.S. Patents, expressly incorporated herein by reference, define aspects of micropayment, digital certificate, and on-line payment systems: 5,930,777 (Barber); 5,857,023 (Demers et al.); 5,815,657 (Williams); 5,793,868 (Micali); 5,717,757 (Micali); 5,666,416 (Micali); 5,677,955 (Doggett et al.); 5,839,119 (Krsul; et al.); 5,915,093 (Berlin et al.); 5,937,394 (Wong, et al.); 5,933,498 (Schneck et al.); 5,903,880 (Biffar); 5,903,651 (Kocher); 5,884,277 (Khosla); 5,960,083 (Micali); 5,963,924 (Williams et al.); 5,996,076 (Rowney et al.); 6,016,484 (Williams et al.); 6,018,724 (Arent); 6,021,202 (Anderson et al.); 6,035,402 (Vaeth et al.); 6,049,786 (Smorodinsky); 6,049,787 (Takahashi, et al.); 6,058,381 (Nelson); 6,061,448 (Smith, et al.); 5,987,132 (Rowney); and 6,061,665 (Bahreman). See also, Rivest and Shamir, "PayWord and MicroMint: Two Simple Micropayment Schemes" (May 7, 1996), expressly incorporated herein by reference; Micro PAYMENT transfer Protocol (MPTP) Version 0.1 (22-Nov-95) et seq,; Common Markup for web Micropayment Systems, (09-Jun-99); "Distributing Intellectual Property: a Model of Microtransaction Based Upon Metadata and Digital Signatures", Olivia, Maurizio.

Page 18, lines 19-Page 19, line 17, please amend as follows:

MARKUP COPY

The OTS™ API 31 is a real time asynchronous API, that supports up to over 400 ports meaning, that to control a large number of phone 33 calls, e.g., a system with 100 lines. A client application 30 may be developed which is controlling the server, and communicates through the basic API 31. However, in this case, it is necessary to develop a multi-threaded application that can moderate the real time asynchronous communication needs of all of those 100 telephone calls simultaneously, which this is complex. Through the preferred DLL 32 mechanism according to the present invention, it is possible to implement the logic required to handle one individual phone 33 call in the form of a DLL 32, using any programming language desired.

For example, the routing of the call may be: the call comes in, some data is received from the telephone network, that data is matched against an external database, conditions in the call center are analyzed, and then based on this analysis, the routing of the call is determined and it is given an instruction to route this call here or play this message to the caller. That call must be multi-thread into the control application, among other call handling needs, to do it through a conventional API. If this is implemented in a DLL 32, the primary call control application can simply, when a call comes in, pass it to the DLL 32, and the DLL 32 is independently invoked simultaneously on as many phone 33 calls that need that DLL 32 at that moment in time. So if 100 calls come in requiring the same logic, as each phone 33 call comes in, a new instance of the DLL 32 is invoked, and it runs independently from the other instances. What this does is to take advantage of some of the internal capabilities of Windows NT, and eliminate the need to actually code the multi-threading and the management of the multiple calls in the logic itself. Tell it has to call that DLL 33 at this point in time.

Page 18, lines 19-Page 19, line 17, as amended:

CLEAN COPY

The OTS™ API 31 is a real time asynchronous API, that supports up to over 400 ports meaning, that to control a large number of phone 33 calls, e.g., a system with 100 lines. A client application 30 may be developed which is controlling the server, and communicates through the basic API 31. However, in this case, it is necessary to develop a multi-threaded application that can moderate the real time asynchronous communication needs of all of those 100 telephone calls simultaneously, which this is complex. Through the preferred DLL 32 mechanism according to the present invention, it is possible to implement the logic required to handle one individual phone 33 call in the form of a DLL 32, using any programming language desired.

For example, the routing of the call may be: the call comes in, some data is received from the telephone network, that data is matched against an external database, conditions in the call center are analyzed, and then based on this analysis, the routing of the call is determined and it is given an instruction to route this call here or play this message to the caller. That call must be multi-thread into the control application, among other call handling needs, to do it through a conventional API. If this is implemented in a DLL 32, the primary call control application can simply, when a call comes in, pass it to the DLL 32, and the DLL 32 is independently invoked simultaneously on as many phone 33 calls that need that DLL 32 at that moment in time. So if 100 calls come in requiring the same logic, as each phone 33 call comes in, a new instance of the DLL 32 is invoked, and it runs independently from the other instances. What this does is to take advantage of some of the internal capabilities of Windows NT, and eliminate the need to actually code the multi-threading and the management of the multiple calls in the logic itself. Tell it has to call that DLL 33 at this point in time.